كتاب هندسه البرمجيات العربي

ادعو لي من فضلكم (ولك من الله بمثلها)

مقدمة في هندسة البرمجيات 1

مقدمة:
هندسة البرمجيات هو عبارة عن فرع من علوم الحاسب الآلي
وظهرت أهميته من بداية ظهور أول برامج... زكان يشمل تحليل النظم وهندسة النظم

وغيرها من التخصصات
ولقد ظهر هذا الاسم لأول مرة سنة

مامعنى هندسة؟
كما ورد في بعض الكتب كانت الهندسة عبارة عن حل المشاكل...
لأن المشكلة هي عبارة عن أساس أي عمل
فإذا اردت عمل مشروع فإنه لديك مشكلة وهي كيفية عمل المشروع
لذا ظهرت كلمة مشكلة ومعالجة المشاكل مصاحبة للهندسة
والمهندس هو الذي يحل المشكل بطرقه العلمية المقننة
يتضمن المعنى العام للهندسة في:
تطبيق الرياضيات والعلوم-
تجميع الحلول-
الاتجاه للمستخدم -
نفع المجتمع -

ما معنى برمجيات ؟
نطلق كلمة برنامج على الـ
Program و توثيقاته معا
ونعني بالتوثيقات أي جميع الوثائق التي تأتي مع البرنامج
من "دليل المستخدم" الى " تصميم النظام" الى "طلبات العملاء" الى "النظرة أو الرؤية" وغيرها من الوثائق المصاحبة للبرنامج

الآن ما معنى هندسة البرمجيات ؟
وصلنا الآن الى السؤال الذي يطرح دائما في كل مكان
ولكن مشكلة السؤال هو انه حتى باللغة النجليزية
لا يوجد تعريف شافي وصريح للمصطلح
ولكن نستطيع ان نقول ان هندسة البرمجيات هي:
تطبيق الافكار الهندسية على بناء البرامج...

أو بصورة أخرى:
هي خطوات منضبطة هندسية تهتم بجميع جوانب انتاج البرامج

تعريف IEEE:
هي التطبيق المنظم والمنضبط والكمي لتطوير وتشغيل و صيانة البرامج.

بماذا نهتم ؟
نهتم بكل النظريات و الطرق والادوات الخاصة بتطوير البرامج بطريقة محترفة

مالفرق بين هندسة البرمجيات وعلوم الحاسب الآلي؟
طبعا علوم الحاسب الالي أشمل و أعم
والهندسة البرمجية هي فرع منه

ماهي العملية البرمجة ؟
هي عبارة عن خطوات يعمل بها المهندسون
لينتجوا لنا برنامجا ناجحا
وخطواتها بوجه عام كالاتي:
- المتطلبات (وتحدد متطلبات النظام او البرنامج)
- التطوير - ويشمل انتاج البرنامج او النظام
- الفحص - ويشمل اختبار المنتج والتاكد من جودته
- الصيانة والارتقاء- ويشمل صيانة البرنامج و قابليته للاستجابة للمتغيرات

ماهي نماذج أو قوالب العمليات الهندسية؟
هي عبارة عن نماذج او قوالب جاهزة
يتخذها المهندس لعمل البرنامج
وهي انواع فمثلا العامة منها:
-نموذج الشلال

- نموذج التطور
- نموذج التجميع واعادة الاستخدام

ما هي فروع هذه الهندسة ؟
فروعها كثيرة....منها
- تصميم الانظمة
- تحليل النظم
- الاختبار والجودة
- هندسة المتطلبات

مصطلحات:
Software Engineering هندسة برامج
Program برنامج
Software برنامج
Engineering هندسة
Software Process العملية البرمجية
Software Process Model نماذج العمليات البرمجية
Evolutionary Model نموذج التطور
Waterfall Model نموذج الشلال
Resuse اعادة الاستخدام
Model نموذج او قالب
Development تطوير
QA & Testing الاختبار والجودة
Requierment المتطلبات

مقدمة في هندسة البرمجيات 2

تكلفة البرامج

عادة تكون تكلفة النظام او البرنامج أكثر من تكلفة العتاد او الجهاز الذي يعمل عليه الربنامج

وتكون التكلفة على الصيانة اكثر مما يصرف على تطويره

وفي بعض الاحيان يكون صيانة البرنامج تعادل اضعاف ما كلف في تطويره

إذا فهندسة البرمجيات تهتم بانتاج برامج

ذات تكلفة فعالة أي اننا ننتج برامجا

بجودة عالية وباقل تكلفة ممكنة

****

تكلفة هندسة البرمجيات

تقريبا 60% من تكلفة البرنامج تذهب في طور التطوير

و 40% تصرف على الاختبارات والفحص

ومرحلة النشئ أي البناء تكلف اكثر من مراحل التطوير

وعادة التكلفة تختلف من برنامج الى آخر

ويعتمد على نوع النظام المراد تطويره وعلى متطلبات صفات ذلك النظام

مثل الآداء و مقدار اعتمادية البرنامج

ويكون توزيع التكاليف على حسب نوع نموذج التطوير اللذي استخدمناه

****

ماهو الـ (Computer-Aided Software Engineering) CASE ؟

هي عبارة عن أدوات تساعد المهندس على

انجاز اعمالا هندسية تساعد في أتمتة ودعم
عمليات عمل البرامج...
وتنقسم الى قسمين عليا وسفلى.
-تستخدم العليا في المراحل الاولى في عمليات البرامج
مثل جمع المتطلبات وتحليلها
- أما السفلى فتستخدم في المراحل المتقدمة من
عمليات البرامج مثل التطوير والبرمجة والاختبار
****

ماهي صفات البرنامج الجيد ؟
أهم شيء في البرامج ان
يوفي بكل المتطلبات التي طلبها الزبون
سواء كانت المتطلبات في الأداء أم الاعمال التي يؤديها البرنامج
والصفات المهمة في جودة البرنامج هي:
- قابليته للصيانة
فيجب ان يكون البرنامج قابل للتعديلات حسب المتغيرات
المصاحبة له في النظام
- اعتمادية البرنامج
يجب ان يكون البرنامج مصمم
بحيث ان الزبون يستطيع ان يعتمد عليه
وتكون مصداقية النتائج واضحة للمستخدم
و يكون مع البرنامج وثيقه تحدد حدوده ومدى مقدرته بصدق
- الكفاءة
يجب ان لايهدر مصادر النظام بلا فائدة
- قابليته للاستخدام
يجب ان يكون البرنامج قابل للاستخدام من قبل المستخدمين
****

كيف نعمل بطريقة صحيحة؟
- أولا بادارة و تخفيف درجة تعقيد النظام
ويكون بفهمنا الجيد للنظام وتكسيره الى انظمة اصغر بحيث نفهمه كاملا
- ثانيا تحويل الاحتمالات الى خطط
- ثالثا ادارة المتغيرات
ويشمل متغيرات المتطلبات
وكذلك متغيرات النظام
- رابعا التحدث مع الزبون
- خامسا تطبيق هندسة البرامج

هذه نظره عامة ليس الا

وسنتكلم عن كل شيء بالتفصيل لاحقا

والان؟؟؟؟

الآن عندنا معلومات عن هندسة البرامج

وماهي اهتماماتهم

وكيف يمشون ويفكرون هؤلاء المهندسون

الجزء الثالث من المقدمة

سيكون استعراضا للنماذج

وخطوات العمل العامة لعمل

برامج محترفة

مقدمة في هندسة البرمجيات 2

تكلفة البرامج

عادة تكون تكلفة النظام او البرنامج أكثر من تكلفة العتاد او الجهاز الذي يعمل عليه الربنامج

وتكون التكلفة على الصيانة اكثر مما يصرف على تطويره

وفي بعض الاحيان يكون صيانة البرنامج تعادل اضعاف ما كلف في تطويره

إذا فهندسة البرمجيات تهتم بانتاج برامج

ذات تكلفة فعالة أي اننا ننتج برامجا

بجودة عالية وباقل تكلفة ممكنة

****

تكلفة هندسة البرمجيات

تقريبا 60% من تكلفة البرنامج تذهب في طور التطوير

و 40% تصرف على الاختبارات والفحص

ومرحلة النشئ أي البناء تكلف اكثر من مراحل التطوير

وعادة التكلفة تختلف من برنامج الى آخر

ويعتمد على نوع النظام المراد تطويره وعلى متطلبات صفات ذلك النظام

مثل الآداء و مقدار اعتمادية البرنامج

ويكون توزيع التكاليف على حسب نوع نموذج التطوير اللذي استخدمناه

****

ماهو الـ (Computer-Aided Software Engineering) CASE ؟

هي عبارة عن أدوات تساعد المهندس على

انجاز اعمالا هندسية تساعد في أتمتة ودعم

عمليات عمل البرامج...

وتنقسم الى قسمين عليا وسفلى.

-تستخدم العليا في المراحل الاولى في عمليات البرامج

مثل جمع المتطلبات وتحليلها

- أما السفلى فتستخدم في المراحل المتقدمة من

عمليات البرامج مثل التطوير والبرمجة والاختبار

****

ماهي صفات البرنامج الجيد ؟

أهم شيء في البرامج ان

يوفي بكل المتطلبات التي طلبها الزبون

سواء كانت المتطلبات في الأداء أم الاعمال التي يؤديها البرنامج

والصفات المهمة في جودة البرنامج هي:

- قابليته للصيانة

فيجب ان يكون البرنامج قابل للتعديلات حسب المتغيرات

المصاحبة له في النظام

- اعتمادية البرنامج

يجب ان يكون البرنامج مصمم

بحيث ان الزبون يستطيع ان يعتمد عليه

وتكون مصداقية النتائج واضحة للمستخدم

و يكون مع البرنامج وثيقه تحدد حدوده ومدى مقدرته بصدق

- الكفاءة

يجب ان لايهدر مصادر النظام بلا فائدة

- قابليته للاستخدام

يجب ان يكون البرنامج قابل للاستخدام من قبل المستخدمين

****

كيف نعمل بطريقة صحيحة؟

- أولا بادارة و تخفيف درجة تعقيد النظام

ويكون بفهمنا الجيد للنظام وتكسيره الى انظمة اصغر بحيث نفهمه كاملا

- ثانيا تحويل الاحتمالات الى خطط

- ثالثا ادارة المتغيرات

ويشمل متغيرات المتطلبات

وكذلك متغيرات النظام

- رابعا التحدث مع الزبون

- خامسا تطبيق هندسة البرامج

هذه نظره عامة ليس الا

وسنتكلم عن كل شيء بالتفصيل لاحقا

والان؟؟؟؟

الآن عندنا معلومات عن هندسة البرامج

وماهي اهتماماتهم

وكيف يمشون ويفكرون هؤلاء المهندسون

الجزء الثالث من المقدمة

سيكون استعراضا للنماذج

وخطوات العمل العامة لعمل

برامج محترفة

مقدمة في هندسة البرامج - الجزء الثالث

نماذج الهندسة البرمجية
سنتكلم عن 4 اشياء مهمة
- دورات الحياة
- لغات النماذج
- العمليات
- الطرق

أولا: دورات حياة المنتج

وهو عبارة عن نموذج. هذا النموذج ينظم سلسة العمليات والنشاطات
في بناء مشروع البرنامج
وهو يعتبر نظرة عامة على جدول المشروع.

النموذج الهندسي
هذا النموذج يتكون من:
- مرحلة الاكتشاف
- مرحلة الاختراع
- مرحلة البناء
وهذه المراحل تتداخل حسب نوع الهندسة

وبما اننا في هندسة البرامج
فستكون المراحل المهمة كمايلي:

- مرحلة المتطلبات
- مرحلة التخصيص
- التصميم العالي أو العام
- مرحلة التصميم السفلي
- التطبيق
- التجميع
-الاختبار
- التسليم

وسوف نتكلم الان عن كل مرحلة بشيء من التفصيل
وكل مرحلة لها مطورين ومختصين وشهادات ومحترفين
وهذا لن تجده الا في الحياة العملية
وليست في النظريات

- مرحلة المتطلبات
وهي أهم مرحلة على الاطلاق
فبدونها المشروع لن ينجح ابدا وهناك عدة شركات تتخصص
في جمع هذه المتطلبات ومن ثم تخصيصها
وللمعلومية ان جميع المتطلبات يؤثر كثيرا في سير المشروع
وبدونها لن يكون المشروع جيدا
ولكي نبسط الموضوع اليكم هذا المثال
" جاء عدد من الموظفين الى الادارة وقالوا نحتاج الى وسيلة مواصلات
واخبروا المهندسون بذلك فقام احد المهندسين ليثبت المشاكل التي تنتج عن
عدم الدقة في جمع المتطلبات
وطلب من احد المبرمجين ان يرسم ما فهم على ورقة
وكرر الطلب نفسه مع أحد المظفين وكذلك مع أحد الاداريين
العبرة تكمن في ان الاداري رسم على الورقة صورة لدراجة هوائية
والمبرمج رسم عربة تجرها خيول
والموظف رسم سيارة"

قد تكون القصة طريفة ولكن الاهم من ذلك كله ان
يتبين لنا أهمية جمع المعلومات
فلو بدا هذا المبرمج في البرنامج وجاء الى يوم التسليم
فيكتشف ان الادارة تخبره ان البرنامج لن يفيدهم بشيء
لانه وبكل بساطة لا يحتاجونه ولا يلبي حاجتهم
مهما كان البرامج جيدا وكاملا
فلن يعرف احد مقدار الجهد والتعب الذي بذله هذا المبرمج

- مرحلة التخصيص
عودة الى المراحل... مع مرحلة المتطلبات
يليها مرحلة التخصيص وهي تحليل واستخلاص
المتطلبات وتصنيفها وتوثيقها وعمل وثيقة مشهورة تعرف
بوثيقة المشروع وتسمى ايضا وثيقة تخصيص متطلبات البرنامج
(Software Requierment Specification (SRS

- مرحلتا التصميم العالي والسفلي
وهنا يتم رسم وتخطيط
كيف يكون شكل المشروع
ويكون التصميم العالي عاما قدر الامكان ويبين فيه
كيف يتكامل المشروع مع الانظمة الاخرى
وكيف يكون شكله العام.
اما التصميم السفلي فيعنى بالتعمق قليلا في
المشروع فيحدد كيف تتكامل المكونات مع بعضها البعض
ويصف الحركات وسيرورة النظام

- مرحلة التطبيق

وهذه المرحلة هي المرحلة المشهورة عنا نحن
معشر المبرمجين
فالناس لا يعرفون غير اننا نبرمج
فهذه المرحلة عبارة عن ترجمة كل المراحل السابقة
الى كود بلغة او عدة لغات وتطبيقها
على الكومبيوتر

- مرحلة الاختبار
وهنا يتأكد المهندس او الذين يعملون على
الجودة في اختبار البرنامج و قياس مدى كفائته

- مرحلة التسليم
وهنا يتم تسليم البرنامج الى الزبون
و يشمل تركيب البرامج ايضا وتقديم
دليل المستخدم

هنا تطرح عادة اسئلة مهمة
وهو هل يجب ان نتبع جميع هذه المراحل؟
وهل هذه المراحل هي الوحيدة ؟
هذه المراحل طويلة فهل نستطيع عمل أكثر من مرحلة في نفس الوقت ؟
واذا كان ذلك هلهناك سير معين في ترتيب هذه المراحل ؟

الاجابة عن هذه الاسئلة سوف يكون في
نماذج دورات الحياة
وهي مجموعة من النماذج
تسير بطريقة معينة وتوزع عليها هذه المراحل

استدراك

دعونا نكمل بعض النواقص التي تركناها وقلنا اننا سنتكلم عنها فيما بعد.

عندما تكلمنا عن دورات الحياة ، ومراحل سير المشروع (راجع الجزء الثالث) وهناك طرحنا هذه الاسئلة:

هل يجب ان نتبع جميع هذه المراحل؟

وهل هذه المراحل هي الوحيدة ؟
هذه المراحل طويلة فهل نستطيع عمل أكثر من مرحلة في نفس الوقت ؟

واذا كان ذلك هلهناك سير معين في ترتيب هذه المراحل ؟

قلنا ان البرامج تمر بمراحل معينة. وقلنا انه بما اننا سنتبع النموذج الهندسي يجب علينا ان نحدد هذه المراحل. و يجب ان يكون معلوما لدينا أن ترتيب المراحل يعتمد تماما على نموذج دورة الحياة. فبعض دورات الحياة تسمح باعادة المراحل. وبعضها تمر عليها مرة واحدة فقط.

أما بالنسبة لتحديد العدد (أي عدد المراحل) فبعض نماذج دورات الحياة يضم مرحلتين او ثلاث في مرحلة واحدة ، وبعضها يفصل المرحلة الواحدة الى مراحل اصغر.

بعد هذا الحديث يتبين لدينا التالي:

وهو ان مراحل وخط سير البرامج ليست محددة لا بعدد ولا بزمن و لا بترتيب ، ولكنها محكومة بنموذج دورة الحياة. وسوف نتكلم الان عن دورات الحياة.

تعريف

مرة أخرى ماهي دورات الحياة ؟
وهي عبارة عن نموذج ، هذا النموذج ينظم سلسة العمليات والنشاطات التي تكون بناء مشروع البرنامج.

وهي أيضا تعتبر نظرة عامة على جدول المشروع.

هنالك أنواع من نماذج دورات الحياة وهذه النماذج قد وضعت من قبل الخبراء في هذا المجال وهو مجال لا يزال قيد التطوير في بيئات العمل المختلفة، وقد تم تجميع هذه النماذج وترتيبها وتعميمها بحيث تناسب أغلب بيئات العمل ، والابحاث مستمرة هناك.

واذا نظرت الى احدى الشركات ستجدهم يعملون على نموذج دورة حياة خاصة بهم ، ولكن الاغلب انهم يستخدمون الدورات القياسية . سنذكر الان 5 نماذج مهمة وعامة ، وتستطيع ان تستخدمها دائما حتى في المشاريع الصغيرة والكبيرة.

أنواع دورات الحياة
1. نموذج الشلال.

2. نموذج التمرحل أو التدرج.
3. نموذج التطور.
4. نموذج التصاعد.

5. نموذج الحلزون.

دورة الحياة الأولى : نموذج الشلال

من المعروف عن الشلال ان يتجه من اعلى الى اسفل ولن يعكس اتجاهه في اي حال من الاحوال.

من هنا يتكون هذا النومذج وهو البسيط جدا

ويستخدم دائما للمشاريع والبرامج الصغيرة.

وهي اننا ننتقل من عملية الى عملية دون ان نكرر اي عملية او نرجع الى عملية سابقة

او حتى نتخطى عملية اخرى.

المراحل العامة المكونة لهذا النموذج

1. تطوير الفكرة :

وهنا مرحلة مهمة وبها تتكون المشاريع وهي ان نأتي بالافكار الاساسية للمشروع وتبدأ بفكرة محددة وهي ان مشروعنا هدفه كذا وكذا ، ونستطيع ان نعمل فيها جسلة العصف الذهني والتي تكلمنا عنها سابقا.(راجع موضوعي في العصف الذهني)

2. المتطلبات :

وهي مرحلة جمع متطلبات البرنامج من الزبائن وجميع المتأثرين بالبرنامج ، ويجب قبل الانتهاء من هذه المرحلة ان يكون عند الفريق تصور كامل للمشروع وهي مرحلة نقوم فيها ايضا بعملية العصف ذهني. ويجب الجلوس مع العميل في هذه الحالة قدر المستطاع وسوف تكون هناك دروس خاصة لجمع المتطلبات لان شرحه يطول وهناك متخصصون في هذا المجال.

3. التصميم العالي او العام: وهنا يتم رسم خطوط عريضة لشكل البرنامج العام.

4. التصميم التفصيلي.

5. التطبيق وكتابة الكود.

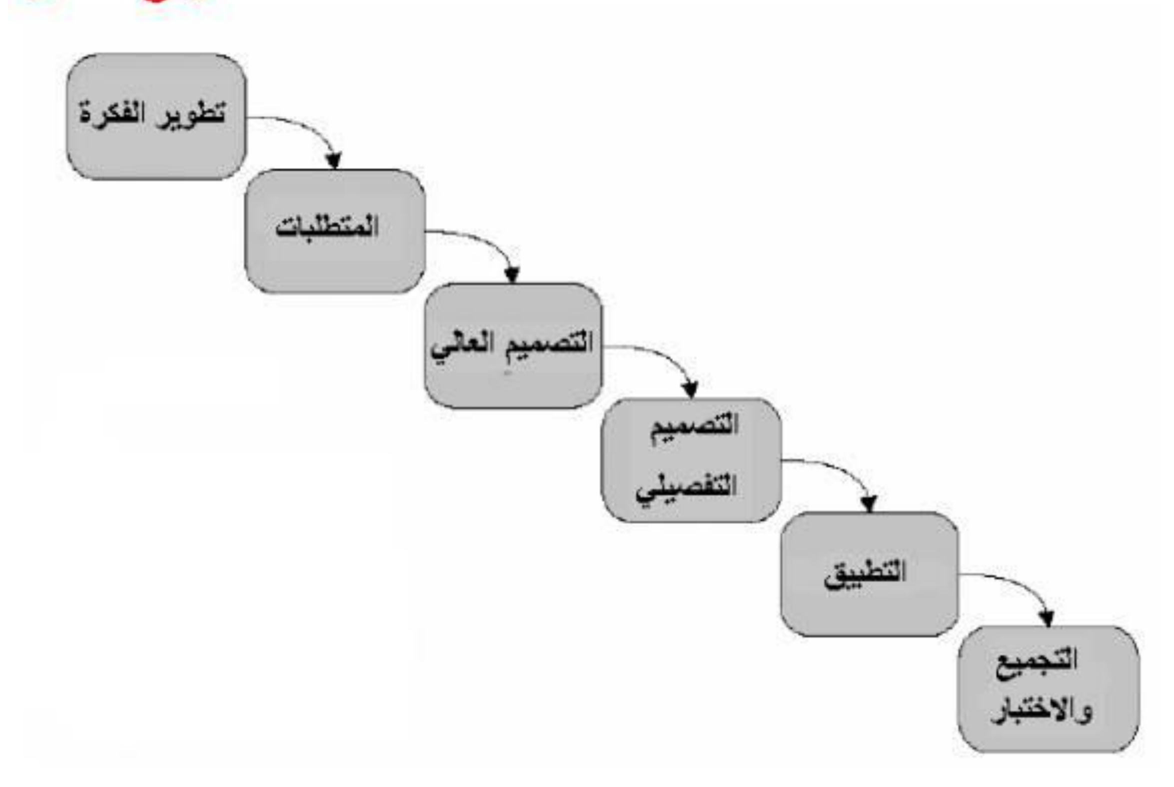
6. التجميع والاختبار.

خطة سير النموذج بهذا الشكل 1 2 3 4 5 6 (أي تسلسلي).

طبعا هناك مشكلة واحدة وهي اذا احتجنا الى اعادة احدى المراحل او العمليات، فماذا نفعل حينها ؟ وهذا يخل بالنموذج الشلال.

في هذه الحالة يوجد حلان : أولهما ان نعيد استخدام هذا النموذج مرة أخرى والثاني ان نستخدم نموذج اخر من البداية.

Error!



نموذج الشلال أشهر دورة حياة للبرامج.

دورة الحياة الثانية : نموذج التدرج

وهو تقريبا نفس نموذج الشلال ومراحله العامة كالاتي :

1. تطوير الفكرة.

2. المتطلبات.

3. التصميم العالي او العام.

4. مرحلة مفصلة تضم :

4.1. التصميم

4.2. التطبيق كتابة الكود.

4.3. الاختبار

4.4. التسليم.

خطة سير النموذج كالاتي : 1 2 3 وبعد ذلك نقسم المشروع الى أجزاء (هذه الخطوة تكون في المرحلة 3) . وكل جزء سيكون في مرحلة رقم 4 والمرحلة الرابعة تكرر عدة مرات حتى ننتهي من المشروع.

مثال للتوضيح :

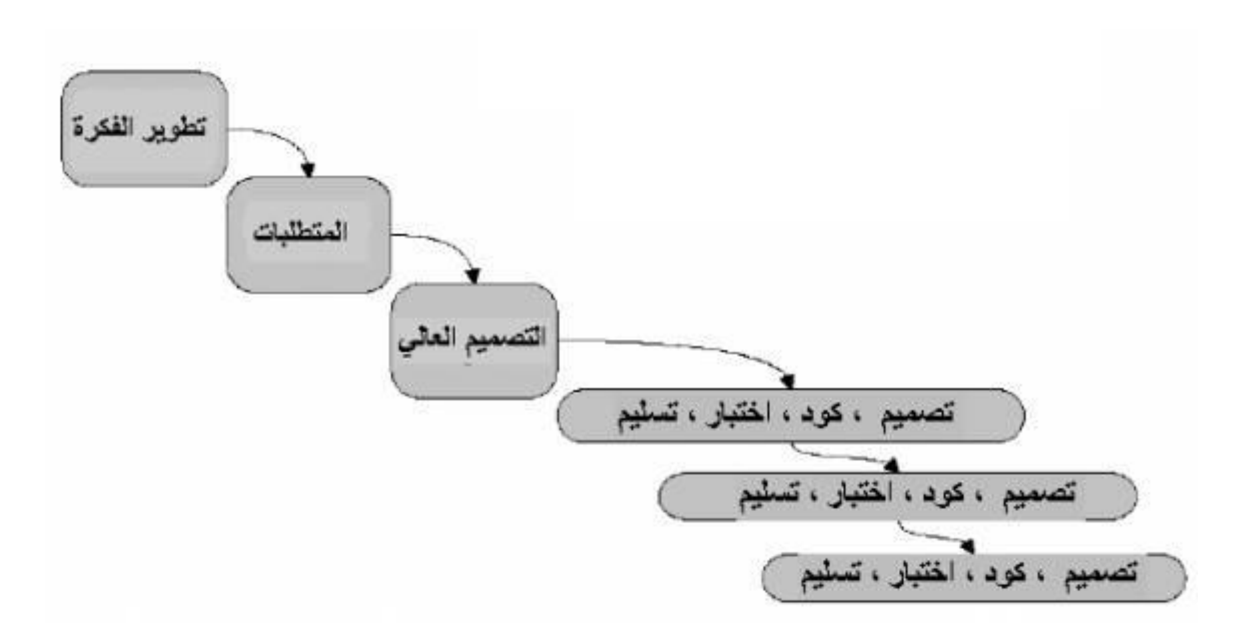
بعدما قسمنا المشروع في خطوة رقم 3 ووضعنا الخطوط العريضة ، نأخذ كل قسم ونصممه، ومن ثم نطبق هذا التصميم ونكتب الكود ، وبعد ذلك نختبره

وبعدها نسلم هذا الجزء. ففي حالة كانت المرحلة ناجحة نبدا في الجزء الذي يليه ولكن اذا لم تنجح يجب اعادة نفس الخطوات داخل المرحلة الرابعة وعدد المرات غير مربوط بشي الا طبعا بوقت المشروع العام.

لو نلاحظ في هذه المرحلة اننا استخدمنا نموذجين من الشلال من 1 الى 3 نموذج ومن 4.1 الى 4.4 نموذجا اخر.

!Error

نموذج التدرج أو التمرحل



. مثال يوضح نموذج التدرج أو التمرحل

مقدمة:

هذا هو الجزء الخامس والاخير من هذه المقدمة ، ولو اني لم اغطي كل شيء بطبيعة الحال . وارد ذلك الى اننا بصدد مقدمة فقط. ولكن نستطيع عمل شيء آخر وهو ان يقوم الاعضاء باختيار موضوع ونتوسع فيه حتى نكتفي ومن ثم نأخذ موضوعا آخر.

استدراك:

كما قلنا أن انواع دورات الحياة

1. نموذج الشلال.
2. نموذج التمرحل أو التدرج.
3. نموذج التطور.
4. نموذج التصاعد.
5. نموذج الحلزون.

وقد غطينا نموذجين والان نتجه الى النماذج الاخيرة....

دورة الحياة الثالثة : نموذج التطور

وهذا النموذج جدا مهم لأنه يستخدم مع الزبائن ذوي الطابع المتقلب. وهم الزبائن الذين يغيرون طلباتهم بشكل مستمر. أو الزبائن الذين لا يعرفون بالضبط ماذا يريدون.

وتكون خطواته الاول مثل الباقين ابتداء من تطوير الفكرة الى التصميم العالي مرورا بالمتطلبات.

وبعد ذلك يبدا الجزء الممتع:

فنبدا بجزء صغير من المشروع وهو الجزء الذي تكون فيه المتطلبات واضحة بالنسبة للفريق والزبون معا.

ونبدا بالتصميم وكتابة الكود و من ثم نختبره فاذا نجح...!!! ننتقل للخطوة القادمة وهي ... اختبار باتجاه المستخدم وهو بكل بساطة عبارة عن جزء كامل من البرنامج يعمل بكفاءة وجاهز للاستخدام ويعطى للمستخدم ليجربه ويعطي تعليقه او ردوده عليه.

وبعد ذلك : تحليل الرد.

وفي النهاية : تحديث المتطلبات لأن المستخدم بعد رؤيته للبرنامج يكون قد تكونت لديه فكرة اوضح عن البرنامج وبالتالي تتغير المتطلبات من متطلبات غامضة الى اخرى اوضح وادق.

بعد ذلك نعود للتصميم السفلي ومن ثم الكود وهكذا.... ولعل هذه الرسمة توضح ما قلناه:

دورة الحياة الرابعة : نموذج التصاعد

وهذا الموذج بسيط جدا ولا يتاج الى شرح كثير. فهنا لا نحتاج الا ان نقسم المشروع الكبير جدا الى مشاريع اصغر فاصغر وبعدها ناخذ كل جزء ونطبق عليه احدى دورات الحياة.

وتكمن اهميته هنا ان المشروع يتصاعد شيئا فشيئا كلما انهينا جزء ركبناه مع باقي الاجزاء بطريقة تصاعدية. أي اننا نكمل الاجزاء الصغير فالاكبر وهكذا. وهذا مفيد جدا اذا كان لديك فريق كبير او مشروع كبير. وفي النهاية نجد ان المشروع اخذ شكله النهائي.

ونستطيع استخدام اكثر من نوع من دورات الحياة ، فقد نستخدم الشلال مع جزء والتطور مع جزء آخر.

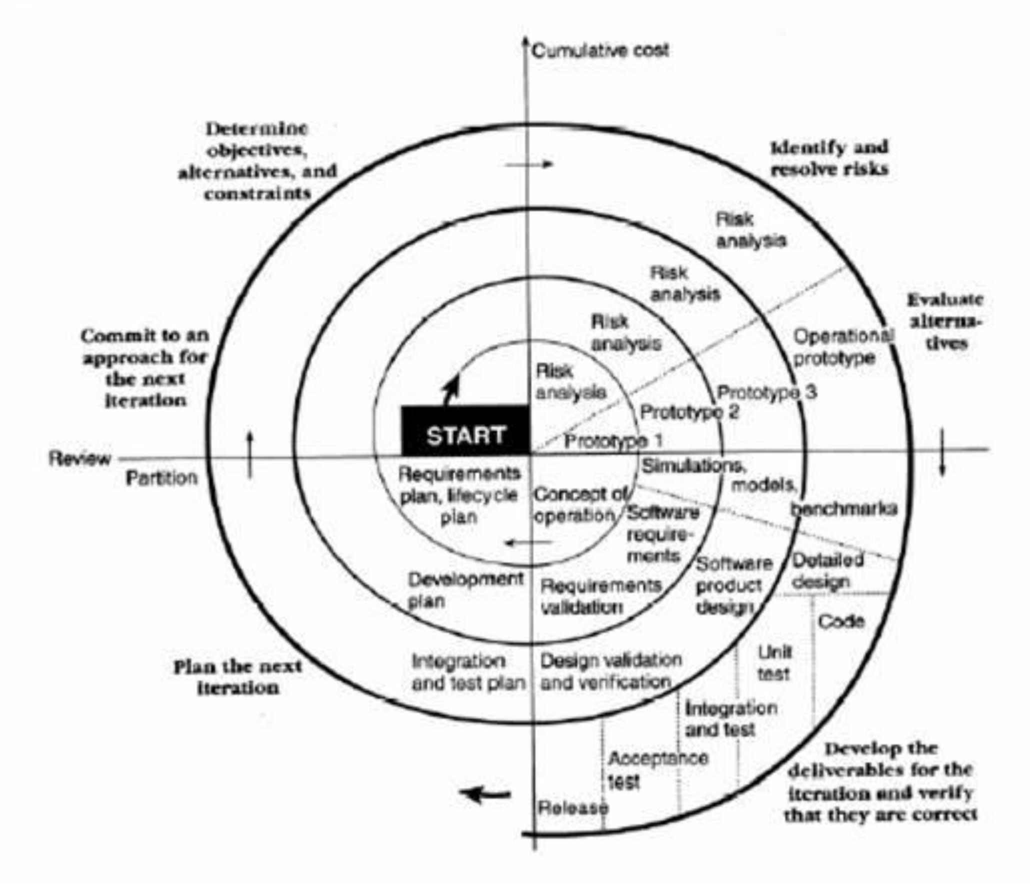
دورة الحياة الخامسة : نموذج الحلزون

وهذا النموذج معقد ومتقدم. يستخدم هذا الموذج في الشركات التي يكون عملها حساسا جدا. ولأنه يشمل مواضيع متقدمة ليس هناك مجال للتوسع فيها الان.

دعونا نرى الرسم التوضيحي (غير مترجم لتعقيده)....

!Error

Spiral



اهم نقطة في هذا النموذج في انه يتكون من نماذج عرض Prototype وهي عبارة عن صور مصغرة للبرنامج (وليست كاملة) تهدف الى اختباره من جهة المستخدم (من الممكن كتابة موضوع منفصل عنه لاحقا).

المهم الان ان الموذج يبدا من نقطة البداية Start ويمر بوجه عام بمراحل مهمة :

المرحلة الاولى:

- تحليل الاخطار (وهذا ايضا موضوع كبير وسأعطي موجز عنه الان)

ومعنى تحليل الاخطار ان فريق العمل يجب ان يضع في حسبانه كل مواطن الاخطار المحدقة بالمشروع ابتداء من الاصابة بفيروسات الى فقدان الاجهزة وسرقتها الى تلف بعضها مرورا بوفاة احد المبرمجين لا قدر الله. ويشمل التحليل وضع خطة بديلة اذا حدث أي من الاخطار المحدقة بالمشروع.

- والان نحدد فكرة العمل.

- وبعد ذلك نبدا بجمع المتطلبات وعمل الخطة واختيار خطة لدورة الحياة.

- بهذا نسلم دمية او نموذج عرض للمستخدم Prototype1 بان هذا هو مفهومنا المبدئي للنظام او المشروع.

المرحلة الثانية:

- نعود ونحلل الاخطار مرة اخرى لاننا استكملنا مجمعة المتطلبات وخاصة بعد ردود المستخدمين.

- جمع المتطلبات (هنا تؤخذ المتطلبات الادق وهي التي تهتم بتفصيل اكثر قليلا) .

- التحقق من المتطلبات (وهنا نتأكد من أننا لا نعارض متطلبات سابقة وانها توافق ما يريده المستخدم).

- و جاء دور خطة التطوير وكيف تسير الخطة البرمجية

- وهنا نكون جاهزين لتسليم الدمية الثانية او نموذج العرض الثاني Prototype2. وهذا النموذج يعمل وليس على الورق وهو تمثيل لما سيكون عليه البرنامج الاصلي.

المرحلة الثالثة:

- و الان نحلل أخطار المرحلة الثالثة.

- نبدأ في عمل النماذج للنظام.

- وبعد ذلك التصميم.

- التحقق من التصميم واختباره.(التاكد ان يتوافق مع المتطلبات)

- عمل خطة التجميع لكل أجزاء التصميم.

- وهنا نكون جاهزين لتسليم الدمية الثالثة او نموذج العرض الثالث Prototype3. وهذا النموذج يعمل وليس وهو نواة البرنامج الاصلي وهو عبارة مجموعة التصاميم العامة او العالية.

المرحلة الرابعة:

- و الان نحلل أخطار المرحلة الرابعة.

- عمل او احضار برامج اختبارات للمنتج.

- التصميم التفصيلي او السفلي او الدقيق.

- طبعا يتبعه كتابة الكود.

- اختبار الوحدة (التي كتبنا كودها وصممناها) ببرنامج الاختبار.

- نبدا في التجميع والاختبار كمجموعة.

- مبروك الان نسلم الدمية الرابعة او نموذج العرض الرابع Prototype4. وهذا النموذج يسمى الدمية التشغيلية.

- ويجرى عليه الاختبار الذي يسمى اختبار الموافقة او القبول من المستخدم.

- والان نسلم البرنامج الى العميل بعد كل هذا العناء.

لو نلاحظ هنا ان هذا النموذج طويل ومعقد ولكنه يوفر تكاليف باهظة في حالة حصول خطاء. لأنه يحوي على اماكن كثيرة جدا لاكتشاف الاخطاء.

هذا وصلى الله وبارك على سيدنا محمد وعلى آاله واصحابه افضل الصلاة وازكى التسليم

مراجعة :

تتكلم هذه المقدمات عن :

مامعنى هندسة؟

ما معنى برمجيات ؟

ما معنى هندسة البرمجيات ؟

بماذا نهتم ؟

ماهي العملية البرمجة ؟

ماهي نماذج أو قوالب العمليات الهندسية؟

ما هي فروع هذه الهندسة ؟

تكلفة هندسة البرمجيات

تكلفة البرامج

ماهو الـ CASE (Computer-Aided Software Engineering) ؟

ماهي صفات البرنامج الجيد ؟

كيف نعمل بطريقة صحيحة؟

نماذج الهندسة البرمجية

دورات حياة المنتج

هذه كانت مقدمة في هندسة البرمجيات....

واعتذر لطولها وتقطعها

ولكن السبب هو انشغالي .....

على العموم اذا رأى أحد انه يستطيع جمعها في موضوع واحد او ملف واح او ينسقه كما يريد فلامانع عندي.....

مع اني اتمنى ان تثبت هذه المواضيع الخمسة ليستفيد الذين ليس عندهم فكرة عن هندسة البرمجيات

لماذا UML ؟ (لما وراء ذلك.....)
المقدمة:
كثير ما نكون منقادين بالتكنولوجيا لاسباب غريبة حقا . ولا نعرف لماذا نستخدم الشيء او حتى لماذا ليس غيره!!!

لذا اجد الكثير من المطورين يعمل البرنامج ومن ثم يكتب الملاحظات، او ينتهي من عمل البرنامج ومن ثم يرسم UML مع الأسف....هذه الظواهر تدل على اننا نستخدم الشيء ونحن لا نعرف لماذا...!!!

لذلك يجب ان يسأل الشخص دائما في كل شيء يتعلمه ثلاث اسئلة:
- ما هو الشيء الذي سأتعلمه؟
- لماذا هذا بالذات ؟
- كيف أتعلم ؟

ودائما هذه الاسئلة تسأل في أغلب الحالات في التعليم والحياة العملية. لأن الشخص الذي يستطيع ان يكتب ولا يعرف لماذا هو يكتب او لماذا تعلم الكتابة؟ فلن يستطيع الانتاج لانه وبكل بساطة يعرف ولا يعرف لماذا يعرف!!!

البداية:
لكي نجيب عن السؤال لماذا UML ؟ يجب ان نجيب على الاسئلة:
- ماهو UML ؟؟
- لماذا UML ؟؟
- كيف نعمل UML ؟؟

ماهو UML ؟سنبدا الان بشيء من التاريخ وهو تاريخ النمذجة لماذا ظهرت النمذجة.... في علم المتطلبات كانت هناك مشاكل قائمة بين المستخدم والمطور على مدى العصور سواء على مستوى الميكانيكا او الكهرباء ومن ثم الحاسب.

كان دائما هناك الفراغ بين المستخدم والمطور. فالمطور بسبب علمه

يعتقد ان الناس من حوله قريبين في مستوى المعرفة والذوق وحتى لو لم يعتقد ذلك فهو لا يقدر حجم المسافة بين الاثنين...

لذا لجأ المهندسون الى وسائل عديدة للتقليل من هذه الفجوة. وبدأت النمذجة بالظهور. فكان المهندس يجمع المتطلبات ويأتي بقطعة تشبه المشروع الكامل للمستخدم ليتاكد ان ما يفعله صحيح قبل الشروع في المشروع الرئيسي.

نعود للحاسب وهندسة البرمجيات. فنحن نطبق القواعد الهندسية على بناء البرامج. ولكن المسألة هنا معقدة للغاية فالبرامج تتجه نحو التعقيد والصعوبة يوما بعد يوم. والحجم يزيد ويتضخم ويمس اعداد اكثر من المستخدمين. لذا نعتقد ان كلما احسن الفريق العمل في جمع المتطلبات كلما زادت نسبة نجاح البرنامج.

لذا كيف نبين فهمنا للمتطلبات (متطلبات المستخدم) ؟
وجدت لغات النمذجة لسببين:
- لنبين مدى فهمنا للمشروع او النظام.
- لتمثيل النظام بلغة دقيقة تفسر تفسيرا واحدا.

اذا فنحن ننمذج النظام لنبين لانفسنا مدى فهمنا للنظام. وبالفعل متى ما عملت نموذج للنظام تستطيع ان ترى مالذي كنت تعنيه عندما نقلت عن المستخدم العبارة "اريد برنامجا سريعا" !!!

ترى هل ماهو سريع بالنسبة لك هو كذلك بالنسبة لي؟؟؟

نعود الان بعدما فهمنا سبب وجود النمذجة نفهم ماهو UML ؟
UML <=== Unified Modeling Language
أو لغة النمذجة الوحدة...

ماذا يعني ذلك؟؟؟ يعني ان هناك لغات توحدت داخل هذه اللغة وانها لغة تختص بالنمذجة. وهذا صحيح فلقد كان المطورون يفكرون بطريقة تسلسلية دالية Functional Orientation .

لتوضيح لذلك. يجب ان يعرف القارئ الفرق بين التصميم لكائني المنحى والدالي المنحنى
Oriented Object Oriented and Functional. وليس هذا مكان للتوسع في هذا الموضوع.
ولكن باختصار عندما كان المهندسون يفكرون بطريقة دالية اي انهم كانوا يقسمون النظام الى عدة اجرائات او دوال وكان البرنامج يعمل بطريقة تسلسلية. فكل البرنامج يدور حول المعلومات و العمليت التي تعمل عليها.

لذا ظهر ما يسمى بالـ DFD (Data Flow Diagram) and ERD (Entity Relationship Diagram) and وهي بالعربية نموذج سير المعلومات و نموذج الكائنات والعلاقات.

بعد ذلك عندما ظهرت اساسيات الكائنية المنحى OO اكتشف المطورين ان الطرق القديمة لا تنفع لنمذجة الانظمة بالطرق التقليدية . وكانت هناك محاولات عديدة مثل OMG وغيرها حتى ظهرت UML لتجمعهم في لغة واحدة وكان بوخ وجاسكوبسون (يعقوب) و رامبو باخراج الابداع الذ نعرف بالـ UML.

لكن وجود UML لا يعني ابدا باي شكل من الاشكال انها الافضل ولكنها مناسبة!!!

الان نعود للسؤال المهم لماذا UML ؟؟

بكل بساطة UML تقدم ثلاث نظرات على النظام:

1- نظرة على العمليات.
2- نظرة على التركيب.
3- نظرة على التصرف.

نظرة على العمليات:

تقدم UML نموذج يدعى Use Case Diagram. هذا النموذج يقدم طريقة لشرح الخدمات والعمليات التي يقدمها النظام وعلاقتها مع المستخدم.
مثال: في نظام التسجيل الجامعي (المستخدم "الطالب" يستطيع تسجيل المادة).

نظرة على التركيب:

تقدم UML نموذج يدعى Class Model. هذا النموذج يقدم طريقة لشرح كيف هي اشكال المعلومات في النظام . وتقدم على شكل "كلاسات" وكل كلاس يحوي صفات و تصرفات.
مثال: في نظام بنكي (كلاس الحساب يحوي الصفات "رقم الحساب و تاريخ فتحه" ويتصرف بعدة طرق مثل "الايداع والسحب").

نظرة على التصرف:

تقدم UML نموذج يدعى State Transition Diagram. وهذا النموذج يشرح النظام على طريقة الحالات State وكل حالة تقود الى حالة اخرى بسبب تحفيز المستخدم للنظام.
مثال: في برنامج منبه للصلاة (يكون البرنامج في حالة خمول يطلب المستخدم "البداية" وينتقل الى حالة العد التنازلي حتى يصل الى الصفر وينتقل الى حالة تشغيل الاذان).

هل يكفي هذا ؟
بالطبع لا!! فلكي تكتمل لغة النمذجة يجب ان نستطيع ان نربط بين النظرات لكي تكتمل عندنا نظرة كاملة على النظام. لذا تقدم UML اشياء اخرى مثل Flow of events and sequence Diagram ليس هنا مجال لذكرها الان.

حتى الان كلام جميل ولكن نبقى ونسأل لماذا UML بالذات ؟
اولا ماذا يعني وجود لغات نمذجة اخرى غير UML ؟ معناه انها صالحة وخناك من يستخدمها ويعرف انها صالحة.
ولكن الاختيار للـ UML يقوم على وجوه عديدة:
- انها لغة تنمذج النظام على ثلاث نظرات لكي تعطي تصور كامل.
- لغة مبنية على اساس كائني المنحى.
- نماذجها سهلة القراءة ولا تعي الا شيئا واحدا دائما (فاذا قرات النموذجمرة او اكثر فستصل دائما لنفس المعنى).
- بقينا ان UML تربط النظرات الثلاث بطرق مهمة بغية التأكد من تكامل هذه النظرات.

عملية حسابية بسيطة:
لو قلنا ان عندنا ثلاث اشخاص كل واحد منهم كلف بنظرة معينة على النظام ومن ثم اجتمعوا ليكونوا النظام الكامل. ولنفرض ان احتمال ان يفقد احدهم متطلبا من متطلبات المستخدم او لا يفهمه هي 0.1
وبما ان الثلاث النظرات لا تعتمد على بعض فيكون عندنا احتمالية ان ننسى او نخطء في فهم متطلب من متطلبات المستخدم هي 0.1 * 0.1 * 0.1 = 0.001.
فما رأيك الان!!! هل عرفت الان لماذا UML.....!!!

بقينا لشيء واحد الان

كيف نتعلم UML؟
هذا ليس موضوع حديثنا الان وله مكانه ان شاء الله ولكن افضل طريقة لتعلمها الذهاب الى كتب المؤلفين (اعني مولفي UML).

ولكن من هنا الى أين ؟
لا أدري ولكن قد تخرج لغات غير UML وتسود. اهم شيء يجب ان نبقى متجهين نحو الافكار والمبادئ وليس التكنولوجيا فالتكنولوجيا تتغير كل دقيقة وتبقى الافكار دائما. فالنمذجة هي المهمة لا يهم انرسمها ام نتلفظ . تتغير الطرق والفكرة واحدة. **لماذا UML ؟ (لما وراء ذلك.....)**
المقدمة:
كثير ما نكون منقادين بالتكنولوجيا لاسباب غريبة حقا . ولا نعرف لماذا نستخدم الشيء او حتى لماذا ليس غيره!!!

لذا اجد الكثير من المطورين يعمل البرنامج ومن ثم يكتب الملاحظات، او ينتهي من عمل البرنامج ومن ثم يرسم UML مع الأسف....هذه الظواهر تدل على اننا نستخدم الشيء ونحن لا نعرف لماذا...!!!

لذلك يجب ان يسأل الشخص دائما في كل شيء يتعلمه ثلاث اسئلة:
- ما هو الشيء الذي سأتعلمه؟
- لماذا هذا بالذات ؟
- كيف أتعلم ؟

ودائما هذه الاسئلة تسأل في أغلب الحالات في التعليم والحياة العملية. لأن الشخص الذي يستطيع ان يكتب ولا يعرف لماذا هو يكتب او لماذا تعلم الكتابة؟ فلن يستطيع الانتاج لانه وبكل بساطة يعرف ولا يعرف لماذا يعرف!!!

البداية:
لكي نجيب عن السؤال لماذا UML ؟ يجب ان نجيب على الاسئلة:
- ماهو UML ؟؟
- لماذا UML ؟؟
- كيف نعمل UML ؟؟

ماهو UML ؟سنبدا الان بشيء من التاريخ وهو تاريخ النمذجة لماذا ظهرت النمذجة.... في علم المتطلبات كانت هناك مشاكل قائمة بين المستخدم والمطور على مدى العصور سواء على مستوى الميكانيكا او الكهرباء ومن ثم الحاسب.

كان دائما هناك الفراغ بين المستخدم والمطور. فالمطور بسبب علمه يعتقد ان الناس من حوله قريبين في مستوى المعرفة والذوق وحتى لو لم يعتقد ذلك فهو لا يقدر حجم المسافة بين الاثنين...

لذا لجأ المهندسون الى وسائل عديدة للتقليل من هذه الفجوة. وبدأت النمذجة بالظهور. فكان المهندس يجمع المتطلبات ويأتي بقطعة تشبه المشروع الكامل للمستخدم ليتاكد ان ما يفعله صحيح قبل الشروع في المشروع الرئيسي.

نعود للحاسب وهندسة البرمجيات. فنحن نطبق القواعد الهندسية على بناء البرامج. ولكن المسألة هنا معقدة للغاية فالبرامج تتجه نحو التعقيد والصعوبة يوما بعد يوم. والحجم يزيد ويتضخم ويمس اعداد اكثر من المستخدمين. لذا نعتقد ان كلما احسن الفريق العمل في جمع المتطلبات كلما زادت نسبة نجاح البرنامج.

لذا كيف نبين فهمنا للمتطلبات (متطلبات المستخدم) ؟
وجدت لغات النمذجة لسببين:
- لنبين مدى فهمنا للمشروع او النظام.
- لتمثيل النظام بلغة دقيقة تفسر تفسيرا واحدا.

اذا فنحن ننمذج النظام لنبين لانفسنا مدى فهمنا للنظام. وبالفعل متى ما عملت نموذج للنظام تستطيع ان ترى مالذي كنت تعنيه عندما نقلت عن المستخدم العبارة "اريد برنامجا سريعا" !!!

ترى هل ماهو سريع بالنسبة لك هو كذلك بالنسبة لي؟؟؟

نعود الان بعدما فهمنا سبب وجود النمذجة نفهم ماهو UML ؟
UML <=== Unified Modeling Language
أو لغة النمذجة الوحدة...

ماذا يعني ذلك؟؟؟ يعني ان هناك لغات توحدت داخل هذه اللغة وانها لغة تختص بالنمذجة. وهذا صحيح فلقد كان المطورون يفكرون بطريقة تسلسلية دالية Functional Orientation .

لتوضيح لذلك. يجب ان يعرف القارئ الفرق بين التصميم لكائني المنحى والدالي المنحنى
Oriented Object Oriented and Functional. وليس هذا مكان للتوسع في هذا الموضوع.
ولكن باختصار عندما كان المهندسون يفكرون بطريقة دالية اي انهم كانوا يقسمون النظام الى عدة اجرائات او دوال وكان البرنامج يعمل بطريقة تسلسلية. فكل البرنامج يدور حول المعلومات و العمليت التي تعمل عليها.

لذا ظهر ما يسمى بالـ DFD (Data Flow Diagram) and ERD (Entity Relationship Diagram) and وهي بالعربية نموذج سير المعلومات و نموذج الكائنات والعلاقات.

بعد ذلك عندما ظهرت اساسيات الكائنية المنحى OO اكتشف المطورين ان الطرق القديمة لا تنفع لنمذجة الانظمة بالطرق التقليدية . وكانت هناك محاولات عديدة مثل OMG وغيرها حتى ظهرت UML لتجمعهم في لغة واحدة وكان بوخ وجاسكوبسون (يعقوب) و رامبو باخراج الابداع الذ نعرف بالـ UML.

لكن وجود UML لا يعني ابدا باي شكل من الاشكال انها الافضل ولكنها مناسبة!!!

الان نعود للسؤال المهم لماذا UML ؟؟
بكل بساطة UML تقدم ثلاث نظرات على النظام:
1- نظرة على العمليات.
2- نظرة على التركيب.
3- نظرة على التصرف.

نظرة على العمليات:
تقدم UML نموذج يدعى Use Case Diagram. هذا النموذج يقدم طريقة لشرح الخدمات والعمليات التي يقدمها النظام وعلاقتها مع المستخدم.
مثال: في نظام التسجيل الجامعي (المستخدم "الطالب" يستطيع تسجيل المادة).

نظرة على التركيب:
تقدم UML نموذج يدعى Class Model. هذا النموذج يقدم طريقة لشرح كيف هي اشكال المعلومات في النظام . وتقدم على شكل "كلاسات" وكل كلاس يحوي صفات و تصرفات.
مثال: في نظام بنكي (كلاس الحساب يحوي الصفات "رقم الحساب و تاريخ فتحه" ويتصرف بعدة طرق مثل "الايداع والسحب").

نظرة على التصرف:
تقدم UML نموذج يدعى State Transition Diagram. وهذا النموذج يشرح النظام على طريقة الحالات State وكل حالة تقود الى حالة اخرى بسبب تحفيز المستخدم للنظام.
مثال: في برنامج منبه للصلاة (يكون البرنامج في حالة خمول يطلب المستخدم "البداية" وينتقل الى حالة العد التنازلي حتى يصل الى الصفر وينتقل الى حالة تشغيل الاذان).

هل يكفي هذا ؟
بالطبع لا!! فلكي تكتمل لغة النمذجة يجب ان نستطيع ان نربط بين النظرات لكي تكتمل عندنا نظرة كاملة على النظام. لذا تقدم UML اشياء اخرى مثل of events and sequence Diagram Flow ليس هنا مجال لذكرها الان.

حتى الان كلام جميل ولكن نبقى ونسأل لماذا UML بالذات ؟
اولا ماذا يعني وجود لغات نمذجة اخرى غير UML ؟ معناه انها صالحة وخناك من يستخدمها ويعرف انها صالحة.
ولكن الاختيار للـ UML يقوم على وجوه عديدة:
- انها لغة تنمذج النظام على ثلاث نظرات لكي تعطي تصور كامل.
- لغة مبنية على اساس كائني المنحى.
- نماذجها سهلة القراءة ولا تعي الا شيئا واحدا دائما (فاذا قرات

النموذجمرة او اكثر فستصل دائما لنفس المعنى).
- بقينا ان UML تربط النظرات الثلاث بطرق مهمة بغية التأكد من تكامل هذه النظرات.

عملية حسابية بسيطة:
لو قلنا ان عندنا ثلاث اشخاص كل واحد منهم كلف بنظرة معينة على النظام ومن ثم اجتمعوا ليكونوا النظام الكامل. ولنفرض ان احتمال ان يفقد احدهم متطلبا من متطلبات المستخدم او لا يفهمه هي 0.1
وبما ان الثلاث النظرات لا تعتمد على بعض فيكون عندنا احتمالية ان ننسى او نخطء في فهم متطلب من متطلبات المستخدم هي 0.1 * 0.1 * 0.1 = 0.001.
فما رأيك الان!!! هل عرفت الان لماذا UML.....!!!

بقينا لشيء واحد الان

كيف نتعلم UML؟
هذا ليس موضوع حديثنا الان وله مكانه ان شاء الله ولكن افضل طريقة لتعلمها الذعاب الى كتب المؤلفين (اعني مولفي UML).

ولكن من هنا الى أين ؟
لا أدري ولكن قد تخرج لغات غير UML وتسود. اهم شيء يجب ان نبقى متجهين نحو الافكار والمبادئ وليس التكنولوجيا فالتكنولوجيا تتغير كل دقيقة وتبقى الافكار دائما. فالنمذجة هي المهمة لا يهم انرسمها ام نتلفظ . تتغير الطرق والفكرة واحدة.

مقدمة

تعتبر هندسة التفاعل البشري Interaction Design فرع من فروع الهندسة أو هندسة النظم ولكننا في علم الحاسب الآلي أدخلناه ضمن ما نحب أن نسميه هندسة البرمجيات. هذه المقالة تتكلم عن جعل تصميم أي شيء بما فيها البرامج قابلا للاستخدام. و يجب أن لا ننسى أن منتجاتنا في النهاية هي للمستخدم. فإذا كان برنامجك قابلا للاستخدام فقد بلغ فائدته المرجوة طبعا بالإضافة إلى عمله الصحيح.

القابلية للاستخدام

لكي نجعل البرنامج قابلا للاستخدام يجب أن نحقق ما يسمى أهداف الاستخدام Usability Goals. هذه الأهداف تضمن لنا جعل المنتج متفاعلا و فعالا و ممتعا بالنسبة للمستخدم. تضمن هذه الأهداف سلامة استخدام المستخدم للمنتج بفعالية لكي يؤدي نشاطاته أو عملياته بطريقة سريعة وفعالة.

الأهداف الستة

- الفعالية effectivenesE.
- الكفاءة Effecincy.
- الأمان Security.
- المنفعة Utility
- قابلية التعلم Learnability.
- قابلية التذكرMomorabilty.

والآن سنفصل في كل هدف من هذه الأهداف بتعريفه ومن ثم شرحه ونلحق كل هدف بمثال توضيحي صغير وفي النهاية نصيحة صغيرة للمصمم.

الفعالية effectivenesE

كيف يكون برنامج فعالا؟

تعرف الفعالية أنها مدى قدرة برنامجك على عمل ما ينبغي عمله.

ما معنى ذلك؟

معناه هنا بسيط وهو أن البرنامج يعمل ما هو مطلوب منه بالضبط بدون زيادة أو نقصان.

مثال:

تخيل لو انك تريد برنامج محاسبة ولكن هذا البرنامج يقدم خدمات جيدة جدا مثلا خطوط جميلة او أداة للرسم!! فهل هذا برنامج فعال؟ بالطبع لا فأنت تريد برنامجا للمحاسبة فماذا يدعم الرسم ويترك المحاسبة.

نصيحة

طبعا قد يكون المثال السابق ساذجا بعض الشيء. ولكن قد تفاجأ لو قلت لك ان الكثير من المبرمجين عند تصميمهم لبرامجهم يذهبون لأشياء يتقنونها جيدا ويبرزونها في برامجهم وقد لاتكون لها علاقة مباشرة بالمنتج ولكن المبرمج يعجبه أن يبرز مهارته في عمل هذه الشيء.

الكفاءة yEffecinc

كيف يكون برنامج ذو كفاءة عالية؟

تعرف الكفاءة على أنها مدى قدرة برنامجك على دعم المستخدم لعمل شيء معين.

ما معنى ذلك؟

معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث تدعم المستخدم في انجاز العمليات التي يريد أن يقوم بها.

مثال:

تخيل لو انك تريد الحصول على معلومة في صفحة انترنت. فكيف تحب أن يكون حصولك عليها؟ هل تفضل الموقع الذي يجعلك تدخل 4 صفحات متتالية للحصول على تلك المعلومة ؟ أم نصف هذا العدد من الصفحات؟ لا حظ أيضا أن المستخدم يحرك المؤشر بين كل صفحة وأخرى ويضر لتتبع كل رابط لكي يصل إليها.

نصيحة
قلل عدد الحركات التي يحتاج عملها المستخدم لعمل شيء ما. و إن لم تسطع فحاول أن تقلل من عناء الحركة المستمرة للمؤشر. فلا تجعل المستخدم يقرأ الصفحة كاملة ويضر الى الذهاب بالمؤشر من آخر الصفحة إلى أولها لكي يضغط على رابط يدعى "للخلف"!!

الأمان Security
كيف يكون برنامج آمناً؟
يعرف الأمان هنا في هذا الموضع بأنه قدرة البرنامج على منع المستخدم من عمل أشياء غير مرغوب فيها أو أشياء فيها خطر إما على المنتج أو على المستخدم.

ما معنى ذلك؟
معناه أن يجب أن يكون في البرنامج شيئين مهمين هما القدرة على منع المستخدم من عمل أخطاء و القدرة على التراجع عن العمل في حالة الخطأ

مثال:
تخيل لو أن لديك برنامجاً يقرأ رسائل البريد مثل الـOutLook ولكنك إذا حذفت الرسائل لا يسألك عن ما إذا كنت متأكدا من هذه العملية. ماذا يكون شعورك في حلة انك ظللت جميع الرسائل وقمت بمسحها خطأ؟ لنأخذ سيناريو آخر وهو أن وأنت في محرر النصوص الخاص بك وكتبت تقرير طويل وبعد ذلك ذهب للقائمة لتضغط على خيار الحفظ وتفاجأ انك ضغط بالخطأ على خيار الإغلاق لأنه بجانبه!!

نصيحة
إذا كان هناك خيارات متاحة للمستخدم ، فحاول منع الخيارات التي لا تناسب الصفحة أو الحالة التي يكون عليها المستخدم. أيضا حاول فصل الخيارات المختلفة وباعد بين الخيارات الخطرة. فباعد مثلا بين خيار الحفظ وخيار الإغلاق. ووفر إمكانية التراجع في البرنامج في حالة قام المستخدم باختيار الخيار خطأ.

المنفعة yUtilit

كيف يكون برنامج ذو منفعة؟
تعرف المنفعة هنا على أنها مدى قدرة برنامجك على توفير الأدوات الممكنة للمستخدم على عمل ما يريد فعله.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث توفر للمستخدم كل ما يحتاج لكي يؤدي العمل الذي يريده بالطريقة التي يريدها.

مثال:
تخيل لو كان برنامج الرسام لا يتيح لك خاصية الرسم الحر!! ومع ذلك فأنت تستطيع أن ترسم أي شيء تريده ولكنك مجبر على استخدام الأدوات المتوفرة فيه لذا فسوف تمل استخدام البرنامج.

نصيحة
دائما عند توفيرك الأدوات في برنامج حاول أن تجعل المستخدم حر في تصرفاته ووفر أدوات مرنة لعمل النشاطات التي يريد عملها.

قابلية التعلم Learnability.
كيف يكون برنامج سهل التعلم؟
تعرف قابلية التعلم على أنها درجة سهولة التعامل مع البرنامج.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث يكون سهل التعامل. من المعروف أن المستخدمين لا يحبون أن يضيعوا أوقاتهم في قراءة دليل المستخدم ويحبون أن يبدؤوا بالعمل مباشرة دون الحاجة إلى تدريب. فهل برنامجك بالسهولة لكي يقوم المستخدم بفهمه من أول 10 دقائق مثلا؟ هل يسطيع أن يعرف ماذا يقدم برنامجك من مجر النظر إليه؟

مثال:
تخيل لو عندك برنامجين يعملان في مجال الكيمياء. فكيف تفضل أن يكون برنامجك؟ هل تفضل البرنامج الذي يوفر مساعدة ظاهرة بجانب البرنامج أم الذي يوفر دليلا للمستخدم؟

نصيحة
ليس دائما يكون البرنامج سهلا بحيث يستطيع المستخدم لذلك فدليل المستخدم الممتع والواضح شيء مهم لبرنامجك. أيضا لا تجعل المستخدم يتلفت يمينا ويسارا لكي يبحث عن شيء ما وبالذات إذا كانت المساعدة.

قابلية التذكرMomorabilty.
كيف يكون برنامج قابلا للتذكر؟
تعرف قابلية التعلم على أنها درجة سهولة تذكر المستخدم كيف يتعامل مع البرنامج.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث إذا عاد المستخدم بعد فترة معينة ليستخدم برنامجك فلا يحتاج الى الكثير من الوقت لتذكر كيف يستخدم البرنامج مرة ثانية.

مثال:
تخيل لو انك تضر للتدريب على برنامج معين كلما انقطعت عنه فترة من الزمن أو أن العمليات التي يقوم بها البرنامج صعبة لدرجة انك تشعر انك تقوم بها لأول مرة.

نصيحة
استخدم الصور الصغيرة للتعبير عن الأشياء فرؤية المستخدم لصورة مغلف رسائل أسرع إدراكا فيما لو كان هناك فقط عبارة أو كلمة "بريد" مثلا. أيضا يجب أن تحاول أن تجعل المستخدم كل مرة يرى فيها الأمر المتاح له يتذكر ماذا يعمل. فليس هناك فائدة في أن ترمز لعمل ما بصورة وهذه الصورة غير مفهومة أو صعبة التذكر.

النهاية
أريد أن أوضح أشياء مهمة:
- قد تكون هذه النصائح عامة ولكنها مفيدة جدا إذا عملت لها اعتبار وأنت تصمم البرنامج.
– احضر من يستطيع ان يقيم تصميمك من ناحية دعمه للاستخدام.
– حاول أن تجعل المستخدم مرتاحا دائما وسعيدا بالعمل في برنامجك.
– لاحظ أن هذه إرشادات لقابلية الاستخدام ويوجد هناك الكثير من الأشياء الأخرى التي ينبغي مراعاتها في التصميم.
– وإذا كان هناك أي أسئلة فأنا مستعد إن شاء الله. مقدمة

تعتبر هندسة التفاعل البشري Interaction Design فرع من فروع الهندسة أو هندسة النظم ولكننا في علم الحاسب الآلي أدخلناه ضمن ما نحب أن نسميه هندسة البرمجيات. هذه المقالة تتكلم عن جعل تصميم أي شيء بما فيها البرامج قابلا للاستخدام. و يجب أن لا ننسى أن منتجاتنا في النهاية هي للمستخدم. فإذا كان برنامجك قابلا للاستخدام فقد بلغ فائدته المرجوة طبعا بالإضافة إلى عمله الصحيح.

القابلية للاستخدام
لكي نجعل البرنامج قابلا للاستخدام يجب أن نحقق ما يسمى أهداف الاستخدام Usability Goals. هذه الأهداف تضمن لنا جعل المنتج متفاعلا و فعالا و ممتعا بالنسبة للمستخدم. تضمن هذه الأهداف سلامة استخدام المستخدم للمنتج بفعالية لكي يؤدي نشاطاته أو عملياته بطريقة سريعة وفعالة.

الأهداف الستة
- الفعالية effectivenesE.
- الكفاءة Effecincy.
- الأمان Security.

- المنفعة Utility
- قابلية التعلم Learnability.
- قابلية التذكرMomorabilty.

والآن سنفصل في كل هدف من هذه الأهداف بتعريفه ومن ثم شرحه ونلحق كل هدف بمثال توضيحي صغير وفي النهاية نصيحة صغيرة للمصمم.

الفعالية effectivenesE

كيف يكون برنامج فعالا؟
تعرف الفعالية أنها مدى قدرة برنامجك على عمل ما ينبغي عمله.

ما معنى ذلك؟
معناه هنا بسيط وهو أن البرنامج يعمل ما هو مطلوب منه بالضبط بدون زيادة أو نقصان.

مثال:
تخيل لو انك تريد برنامج محاسبة ولكن هذا البرنامج يقدم خدمات جيدة جدا مثلا خطوط جميلة او أداة للرسم!! فهل هذا برنامج فعال؟ بالطبع لا فأنت تريد برنامجا للمحاسبة فماذا يدعم الرسم ويترك المحاسبة.

نصيحة
طبعا قد يكون المثال السابق ساذجا بعض الشيء. ولكن قد تفاجأ لو قلت لك ان الكثير من المبرمجين عند تصميمهم لبرامجهم يذهبون لأشياء يتقنونها جيدا ويبرزونها في برامجهم وقد لاتكون لها علاقة مباشرة بالمنتج ولكن المبرمج يعجبه أن يبرز مهارته في عمل هذه الشيء.

الكفاءة yEffecinc
كيف يكون برنامج ذو كفاءة عالية؟
تعرف الكفاءة على أنها مدى قدرة برنامجك على دعم المستخدم لعمل شيء معين.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث تدعم المستخدم في انجاز العمليات التي يريد أن يقوم بها.

مثال:
تخيل لو انك تريد الحصول على معلومة في صفحة انترنت. فكيف تحب أن يكون حصولك عليها؟ هل تفضل الموقع الذي يجعلك تدخل 4 صفحات متتالية للحصول على تلك المعلومة ؟ أم نصف هذا العدد من الصفحات؟
لا حظ أيضا أن المستخدم يحرك المؤشر بين كل صفحة وأخرى ويضر لتتبع كل رابط لكي يصل إليها.

نصيحة
قلل عدد الحركات التي يحتاج عملها المستخدم لعمل شيء ما. و إن لم تسطع فحاول أن تقلل من عناء الحركة المستمرة للمؤشر. فلا تجعل المستخدم يقرأ الصفحة كاملة ويضر الى الذهاب بالمؤشر من آخر الصفحة إلى أولها لكي يضغط على رابط يدعى "للخلف"!!

الأمان Security
كيف يكون برنامج آمناً؟
يعرف الأمان هنا في هذا الموضع بأنه قدرة البرنامج على منع المستخدم من عمل أشياء غير

مرغوب فيها أو أشياء فيها خطر إما على المنتج أو على المستخدم.

ما معنى ذلك؟
معناه أن يجب أن يكون في البرنامج شيئين مهمين هما القدرة على منع المستخدم من عمل أخطاء و القدرة على التراجع عن العمل في حالة الخطأ

مثال:
تخيل لو أن لديك برنامجاً يقرأ رسائل البريد مثل الـOutLook ولكنك إذا حذفت الرسائل لا يسألك عن ما إذا كنت متأكدا من هذه العملية. ماذا يكون شعورك في حلة انك ظللت جميع الرسائل وقمت بمسحها خطأ؟
لنأخذ سيناريو آخر وهو أن وأنت في محرر النصوص الخاص بك وكتبت تقرير طويل وبعد ذلك ذهب للقائمة لتضغط على خيار الحفظ وتفاجأ انك ضغط بالخطأ على خيار الإغلاق لأنه بجانبه!!

نصيحة
إذا كان هناك خيارات متاحة للمستخدم ، فحاول منع الخيارات التي لا تناسب الصفحة أو الحالة التي يكون عليها المستخدم. أيضا حاول فصل الخيارات المختلفة وباعد بين الخيارات الخطرة. فباعد مثلا بين خيار الحفظ وخيار الإغلاق. ووفر إمكانية التراجع في البرنامج في حالة قام المستخدم باختيار الخيار خطأ.

المنفعة yUtilit

كيف يكون برنامج ذو منفعة؟
تعرف المنفعة هنا على أنها مدى قدرة برنامجك على توفير الأدوات الممكنة للمستخدم على عمل ما يريد فعله.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث توفر للمستخدم كل ما يحتاج لكي يؤدي العمل الذي يريده بالطريقة التي يريدها.

مثال:
تخيل لو كان برنامج الرسام لا يتيح لك خاصية الرسم الحر!! ومع ذلك فأنت تستطيع أن ترسم أي شيء تريده ولكنك مجبر على استخدام الأدوات المتوفرة فيه لذا فسوف تمل استخدام البرنامج.

نصيحة
دائما عند توفيرك الأدوات في برنامج حاول أن تجعل المستخدم حر في تصرفاته ووفر أدوات مرنة لعمل النشاطات التي يريد عملها.

قابلية التعلم Learnability.
كيف يكون برنامج سهل التعلم؟
تعرف قابلية التعلم على أنها درجة سهولة التعامل مع البرنامج.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث يكون سهل التعامل. من المعروف أن المستخدمين لا يحبون أن يضيعوا أوقاتهم في قراءة دليل المستخدم ويحبون أن يبدؤوا بالعمل مباشرة دون الحاجة إلى تدريب. فهل برنامجك بالسهولة لكي يقوم المستخدم بفهمه من أول 10 دقائق مثلا؟ هل يسطيع أن يعرف ماذا يقدم برنامجك من مجر النظر إليه؟

مثال:
تخيل لو عندك برنامجين يعملان في مجال الكيمياء. فكيف تفضل أن يكون برنامجك؟ هل تفضل البرنامج الذي يوفر مساعدة ظاهرة بجانب البرنامج أم الذي يوفر دليلا للمستخدم؟

نصيحة
ليس دائما يكون البرنامج سهلا بحيث يستطيع المستخدم لذلك فدليل المستخدم الممتع والواضح شيء مهم لبرنامجك. أيضا لا تجعل المستخدم يتلفت يمينا ويسارا لكي يبحث عن شيء ما وبالذات إذا كانت المساعدة.

قابلية التذكرMomorabilty.
كيف يكون برنامج قابلا للتذكر؟
تعرف قابلية التعلم على أنها درجة سهولة تذكر المستخدم كيف يتعامل مع البرنامج.

ما معنى ذلك؟
معناه أنه يجب أن تصمم برنامجك بطريقة معينة بحيث إذا عاد المستخدم بعد فترة معينة ليستخدم برنامجك فلا يحتاج الى الكثير من الوقت لتذكر كيف يستخدم البرنامج مرة ثانية.

مثال:
تخيل لو انك تضر للتدريب على برنامج معين كلما انقطعت عنه فترة من الزمن أو أن العمليات التي يقوم بها البرنامج صعبة لدرجة انك تشعر انك تقوم بها لأول مرة.

نصيحة
استخدم الصور الصغيرة للتعبير عن الأشياء فرؤية المستخدم لصورة مغلف رسائل أسرع إدراكا فيما لو كان هناك فقط عبارة أو كلمة "بريد" مثلا. أيضا يجب أن تحاول أن تجعل المستخدم كل مرة يرى فيها الأمر المتاح له يتذكر ماذا يعمل. فليس هناك فائدة في أن ترمز لعمل ما بصورة وهذه الصورة غير مفهومة أو صعبة التذكر.